

DOCUMENT RESUME

ED 156 201

IR 006 434

AUTHOR Brockmeier, Richard
TITLE Report on a Highly Used Computer Aid for the Professor in his Grade and Record Keeping Tasks.
PUB DATE Jun 77
NOTE 9p.; Paper presented at the Conference on Computers in the Undergraduate Curricula (East Lansing, Michigan, June 20-22, 1977); For entire proceedings, see IR 006 142

EDRS PRICE MF-\$0.83 HC-\$1.67 Plus Postage.
DESCRIPTORS Academic Records; College Teachers; *Computer Programs; Data Bases; *Electronic Data Processing; Grades (Scholastic); Higher Education; *Management Information Systems; Physics Teachers; *Recordkeeping; *Student Records
IDENTIFIERS *Computer Managed Instruction; Hope College MI; SPARS

ABSTRACT SPARS is a computer data base management system designed to aid the college professor in handling his students' grades and other classroom data. It can handle multiple sections and labs, and allows the professor to combine and separate these components in a variety of ways. SPARS seeks to meet the sometimes competing goals of simplicity of use and high security. The computer package is designed to operate in both batch and interactive modes on a Xerox Sigma 6 computer requiring 14K words of memory. The program is structured with an executive which recognizes commands and then branches into the appropriate routine to handle the request. For clarity, commands can be grouped (according to type of function) into the several different classes. Tables list the individual commands by class with a brief description of the purpose of each.
 (Author/CHV)

 * Reproductions supplied by EDRS are the best that can be made *
 * from the original document. *

ED156201

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

REPORT ON A HIGHLY USED COMPUTER AID FOR THE
PROFESSOR IN HIS GRADE AND RECORD KEEPING TASKS

BY

Richard Brockmeier

"PERMISSION TO REPRODUCE THIS
MATERIAL HAS BEEN GRANTED BY

Ted Sjoersma

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC) AND
USERS OF THE ERIC SYSTEM."

I R 006 434

Report on a Highly Used Computer Aid for the Professor in his Grade and Record Keeping Tasks

Richard Brockmeyer
Hope College
Holland, Michigan 49423
(616) 392-5111 Ext. 2308

Introduction

A data base management system called SPARS for entering and processing a professor's student data has been developed in the physics department at Hope College. The package is designed to operate in both batch and interactive modes on Xerox Sigma 6 computer requiring 14K words of memory. It is the purpose of this paper to describe the design philosophy and some of the features of this package. The results of our experience with its use at Hope College and future plans are to be discussed.

Design of SPARS

SPARS is a data base manager and processor package designed to aid the professor in handling his students' grades and other classroom data. It can handle multiple sections, labs, etc., and allow the professor to combine and separate these components in a variety of ways. It seeks to meet the sometimes competing goals of simplicity of use and high security.

The limits on the number of independent classes per professor, number of students per class, and data per student are dynamic and can be changed as the professor's needs change; the absolute limits are generous enough not to be of concern to the professor (i.e., up to 1,000 classes per professor, up to 5,000 students per class, and up to 256 words per student). The professor references the class by a class code, the student by student number, and data elements within each student's record by means of one to three char-

acter variable-names which the professor designates at the time he creates the variable. Information about these data elements is stored in a Variable Dictionary automatically by SPARS and includes a description of the variable, what width of field should be used whenever it is printed out, and what type of conversion to use. The professor retains control of the spacing and placement of items in his printouts, but is relieved of chores of formatting. The data itself is contained in student records. These facts however are transparent to the professor. He does not need to know the name of the file, its password, or its structure. He merely LOGS-ON to SPARS and identifies his class-code together with some security identification procedures.

The program is structured with an executive which recognizes commands and then branches into the appropriate routine to handle the request. For clarity, the commands can be grouped (according to type of function) into the several different classes which are shown in Table 1. Tables 2 through 9 then show the individual commands within each of the classes together with a brief description of the purpose of the command. In addition to the information presented in the tables, it should be pointed out that all commands begin with either an "*" or a "?". All commands beginning with the "*" perform some operation, whereas all commands beginning with "?" perform no operations but merely give extended on-line documenta-

tion to the user. For example, for every *-command, there is a companion ?-command explaining the *-command. In addition, many error messages will refer the user to a ?-command for a more lengthy description of what the problem is. SPARS consists of 41 *-commands and about 1,000 lines of documentation accessible through ?-commands.

We shall now briefly itemize a representative sample of some of the features which have been designed into SPARS.

1. The user does not need to explicitly create a variable entry in the Variable Dictionary prior to putting data into the variable. SPARS will create the new entry automatically based on whatever information it already has on the variable. The user can later modify any of the parameters with the *DEFINE command if he is unsatisfied with the automatic result.

2. Unlike the storage of conventional FORTRAN variables, prior to the storage of any data in a SPARS variable, the location is set = to a "value" of ?. (e.g., if you requested SPARS to printout the values of test scores, you would get question marks printed out for any students you skipped over.) Further, if the user so wishes, he can enter a "blank" into a variable location for a student (to indicate that a student didn't take a test, etc.). Any subsequent processing such as averages, correlations, etc., ignores "blanks" and "question-marks." Hence, the problem of dealing with missing data is handled automatically for the user.

3. Unlike conventional FORTRAN, one of the DATA-TYPES is the G-type conversion which accepts letter-type grades (A,A-,B+, etc.) and converts them for internal storage as numeric values (40, 37, 33, etc.). Hence, all processing commands such as *AVERAGE, *CORRELATION, *WAVG (weighted average) proceed in a normal fashion. In fact, this allows an easy means for combining both numeric scores and letter-type grades for arriving at a final raw score. Upon output, G-conversion converts the numeric results back into a letter grade in the printout.

4. As a convenience for the professor (and also as a security feature), SPARS automatically enters into the Variable Dictionary the time, the date, and a code indicating via which command a variable was created. It also enters the time, date, and code for any changes which were made to the data stored under each variable.

5. The professor can request his class printouts to occur in any sorted order of any of the variables, nested up to three deep. Submitting the *ORDER command produces a chaining pointer which is stored in the student's record as a transparent new variable which the user cannot reference directly. If one submits the *WRITE command, SPARS will use the transparent chaining variable which was created or referred to by the most recent *ORDER command to produce the printout of the class in the desired order. Any number of sorts can be stored (restricted only by the 256 maximum limit to the number of variables). Hence, future requests for a previously sorted order will not require a fresh sort. However, any change of data which affects the order of any existing sort flags it as not valid, forcing a new sort at the next time it is requested and the transparent variable is deleted automatically the next time the user enters the *DELETE command. Thus, the professor can get multiple copies of his class—by seating chart, alphabetical by name, from highest to lowest on any test, by lab partner in the lab, etc.

Whenever the professor wants to operate on the data, he must communicate to SPARS a list of variable names to be processed or printed out. A convention under SPARS requires that parentheses initiate and terminate the list. The individual names are to be separated by commas or blanks. Two variable names separated by a hyphen indicate all existing variables which lie between the two in an alphabetical sense. Thus, if he wants to add the scores of six quizzes and ten lab experiments, and put the results in variable RAW, he would say: *ADD (Q1-Q6,E1-E10) (RAW).

Security Provisions

The computer facility is operated in such a fashion at Hope College so as to provide its students with as much experience and direct contact with the computer as possible on a modest budget. Students can frequently be found in the operating room. In fact, all evening operations are conducted with student help with no college personnel at the center. Some thought must be given to security. The philosophy followed has been that: (1) the name, account, and password of the file containing the student data will become known, (2) a User Manual for SPARS will accidentally be left lying around by a user, (3) there will be those who will be "curious," and (4) playing games with students should be avoided. At the same time, the system must be sufficiently simple so that the professor with no prior computer experience will not be discouraged from using it. Moreover, SPARS must contain its own security system, and the source should be highly protected.

Such features as encryption of bits in the record words and extensive indirect addressing of data elements are used to mix up the files, but add no complications to ease of use from the user's point of view. Therefore, one can not make sense of the files except through SPARS itself. Additionally, the system knows which users and user accounts are valid, can tell if a file has been copied and then accessed from a foreign (but still valid) account, etc. Means for recording such occurrences are included without letting the unauthorized person know he has been detected. The professor has the option of causing the sign-on protocol to change if added security is desired.

Experience with SPARS

There has been a steady growth in the use of SPARS since its inception at Hope College. At present, approximately 8-10 percent of the student credit hours are processed by professors through SPARS at Hope College. It was developed and initially used solely in the physics department by faculty who were

highly proficient in using the computer. More recently, its use has spread to professors who have had no prior experience with computers. Also, secretaries have quickly learned how to use a terminal and have entered data for professors. This has proven to be a positive initial experience with the computer—something which several confessed was a surprise. This did not come without a number of earlier unsatisfactory experiences, however.

An earlier version of SPARS lacked some of the present automatic features and didn't have on-line documentation. Several professors with no prior computer experience attempted to use this earlier version. Although they entered with enthusiasm, they soon became stuck on one minor point after another, and ultimately surrendered to a feeling of frustration and wasted time.

But the experience was a valuable one. By watching several non-computer users who were willing to allow you to observe their interactions with the package, this writer was able to make some major changes which very significantly improved the package.

The package is now used in the departments of physics, computer science, chemistry, biology, psychology, sociology, foreign languages, English, business and economics. Professors without prior computer experience tend to restrict their use to the rather basic commands which parallel procedures one would follow in manual grade-keeping. Many professors find that just doing this much on SPARS is of value for classes of twenty-five or more students.

Initially, they tend not to grasp the flexibility which computer data management under SPARS can bring to the process of evaluating one's students. With repeated use, many are beginning to discover its capabilities for doing new analyses like computing, z-scores, studying correlations, etc. In the process, this has brought changes to the way some courses are taught and managed. In the physics department, we use SPARS quite extensively to handle the beginning physics sections and labs. SPARS is designed to receive grades from multiple graders, normalize the scores

and sort them out. In addition, it has been used to aid in selecting students for the honors section.

Conclusions

It is our experience at Hope College that professors with no prior computer experience will use the computer when grade handling packages such as SPARS are made readily available and are sufficiently flexible and easy to learn and use. The use of SPARS at Hope College appears to be steadily growing without administrative push or encouragement. In seeking ways of developing the computer as an aid in the process of education, the major emphasis has been placed on interactions between the *student* and the computer in CAI, computer generated tests, self-paced study, etc. Our experience with SPARS at Hope College indicates that efforts to interface the *professor* with the computer in the more traditional classroom and laboratory environment has not yet been fully explored. It may also be true that by giving a professor an easy and flexible tool like SPARS for intermixing grades and scores from a variety of sources including the traditional ones, we may be smoothing the way for his adoption of CAI units and other forms of computer aids. This seems probable for two reasons: (1) Through SPARS he has developed a comfortable familiarity with the computer himself, and (2) with SPARS he has an easy and convenient way for intermixing grades, scores, and information from a variety of sources including the traditional ones.

The next step at Hope College, should the use of SPARS continue to grow, will be to develop additional commands and more automatic procedures for setting up class files with the names and student numbers at the beginning of the semester so that SPARS can be initiated by the professor via a simple command. Finally, some thought is being given to adding a feature which would enable the professor to enter a flag into the Variable Dictionary indicating his authorization to use the contents of that variable for student advising purposes. These variables would contain, for example, information on the student's most recent progress in class. Another package could be designed to search the SPARS files to identify students, particularly freshmen, who are beginning to have academic difficulty so that they could be placed in contact with their advisors prior to midterm. Hopefully, one could get these students into a more suitable class schedule while it is still possible, and save them from losing a semester or becoming dropouts.

Since SPARS is written in FORTRAN, the possibilities of it being transportable to other sites are good with additional work. Some features of SPARS depend upon the word size being thirty-two bits or greater. Other features depend upon options available under the CP-V operating system of XEROX computers which would have to be supplied in some fashion on a different machine. If outside interest develops, the possibilities of transfer to other machines could be explored.

Table 1

CLASSES OF COMMANDS

ASSISTANCE COMMANDS	(Includes on-line documentation for the user)
CONTROL COMMANDS	(These are global commands which affect operation of other commands)
DATA INPUT COMMANDS	(Variety of means for entering data)
DATA OUTPUT COMMANDS	
HOUSEKEEPING COMMANDS	
PROCESS DATA AND PRINTOUT RESULTS COMMANDS	
PROCESS DATA AND STORE RESULTS COMMANDS	
SECURITY COMMANDS	

Table 2

ASSISTANCE COMMANDS

?COMM[ANDS]:	Gives the names of all existing commands.
*GLOS[SARY]:	User can search the DICTIONARY of variables he has created and related parameters.
?HELP:	Lists other ONLINE sources of help.
?NEW:	Initiates a self-study program for learning SPARS.
?SUGG[ESTION-BOX]:	User can direct questions/suggestions to the SPARS administrator.
?xxxx:	Gives documentation about command *xxxx. See text.

Table 3

CONTROL COMMANDS

*DATS[W]:	For non-standard unadvertised features.
*ECHO:	Used in BATCH; toggles flag controlling printout of input.
*INCL[UDE-IF]:	Restricts all processing and printing to a subset of the class of students which user specifies until next *INCL command is received.
*LENG[TH].*WIDT[H]:	Allow user to change page lengths or widths from default values.
*LPRI[NTER]:	While in interactive mode, user can automatically direct all output (except prompts, error messages, and advisories) to the high-speed printer to reduce terminal time.
*ORDE[R]:	User specifies student order of printout of all reports following this command until a new *ORDER is received. It is capable of sorting on any variable in file with nesting up to three levels.
*TOSS[OUT]:	The user may control which data for students are to be ignored (not included) by the actions of all PROCESSING COMMANDS which follow this command.
*TITL[E]:	Enters titles for reports.
*WARN:	User arms/disarms the warning messages for (1) keying-in of non-existent student numbers and (2) whenever user enters data on-top-of existing data.

Table 4

DATA INPUT COMMANDS

*CHAN[GE]:	For entering/correcting the datum stored in a variable for one student.
*GLAB:	For submitting data from student lab section grades. Used with the *LAGD command.
*PROM[PT]:	User enters data for students without need for keying in student numbers. User can request that he be prompted by student's name and that the student order be the same as any previously printed output.
*QUIZ:	Designed for entry of data from "quiz cards." Brief statistics are reported at the end.
*READ:	For entering multiple variable data keyed on the student number.
*SAME:	Similar to *QUIZ except no statistics are given.

Table 5

DATA OUTPUT COMMANDS

*LAGD:	Produces a deck of cards by student and by lab-experiment (or quiz, etc.—the professor specifies). Then the lab graders report the grades back to the professor via these cards. The professor submits these cards in BATCH after the *LAGD command.
*PUNC[H]:	Allows professor to send data to another file or peripheral such as the card punch.
*WRIT[E]:	Prints class reports on variables specified by professor. The order of the students in printout will be controlled by the most recently submitted *ORDER command. The layout of the variables across the page is controlled by the contents of the VARIABLE DICTIONARY and in the order requested by the professor. Extra spaces between fields are generated if the professor places extra spaces between the variable names. Which students are included in the report is controlled by the most recently received *INCLUDE-IF command. Multiple reports from just one command are possible.

Table 6

HOUSEKEEPING COMMANDS

*CHAN[GE]:	Enables professor to correct any entry for any student. This includes the status variable S which if given a value other than 1 will logically drop the student from the class file (student can be retrieved by resetting S to 1).
*DEF[INE]:	Enables the professor to modify contents of the VARIABLE DICTIONARY such as variable definitions and field widths. He may also create new variables in the DICTIONARY for the first time with this command; however this is unnecessary as creation is automatic upon first reference elsewhere in the package.
*DELE[TE]:	Deletes "dead-wood" variables from the file. Deletes transparent variables (see text).
*GLOS[SARY]:	User can search the VARIABLE DICTIONARY and get data on all the SORTS which are current.
*STU[DENTS]:	The professor uses the command to handle any late student additions to his class file.

Table 7

PROCESS DATA AND PRINTOUT RESULTS COMMANDS

*AVER[AGE]:	Computes and prints out class averages and standard deviations of variables requested. (subsections of class calculations possible if used in conjunction with *INCLUDE-IF COMMAND)
*CORR[ELATION]:	Prints out a correlation matrix between any desired variables for entire class (or subset with use of *INCLUDE-IF command).
*TOTA[L]:	Computes class totals of variables requested (or for subset of class with use of *INCLUDE-IF command).

Table 8

SECURITY COMMANDS

*BATCH:	Passwords are not used in BATCH operation. SPARS will not run in BATCH unless the professor first opens BATCH up with the *BATCH command (via timesharing). Professor specifies how many jobs are to be let through in BATCH.
*LOCK:	Enables professor to enter or change his internal SPARS password.

Table 9

PROCESS DATA AND STORE RESULTS COMMANDS#

*ADD:	Adds up scores specified by the professor and stores the result where he specifies.
*ARIT[HMETIC]:	Multiplies, divides, and adds constant value to the value of a variable and stores result in variable specified.
*DROP[-WORST-SCORES]:	Same as *ADD except SPARS will drop the N-worst scores.
*GRAD[E]:	Cause SPARS to assign a grade to each student based on 11 specified threshold values of a specified raw score (i.e., grades on an absolute scale).
*NORM[ALIZE]:	This enables professor to remove and/or detect some of the grading inconsistencies between different labs or recitation sections. It produces a normalized score.
*RANK:	Determines the rank of each student for any variable and stores it in a specified variable.
*WAVG:	Weighted Average Command. Will compute the weighted average of scores and grades.
*ZGRA[DE]:	This command is equivalent to running *ZSCORE followed by *GRADE and then deleting the intermediate result. This computes a grade based on a curve specified by the professor using a raw score and placing the letter grade where specified.
*ZSCO[RE]:	Computes the ZSCORE (indicates how far a student is above or below class average in terms of standard deviation units) based on the raw score specified and stores result in a variable the professor specifies.

#The operation of each of these commands can be caused to act on any subset of the class desired through the use of the *INCLUDE-IF command. The *TOSS command can also be used to affect the operation of these commands.